PPTP
Portable Precision Time Protocol

Aurelio Colosimo
Alessandro Rubini
Overview

PTP (Precision Time Protocol) is a standard for clock synchronization of computers and electronic appliances to be achieved through an ethernet network.

It was defined by the IEEE 1588-2002 standard and revised in IEEE 1588-2008, also known as PTP2.

Our work focuses on PTP2 standard.
Aim

- Provide a unified source code for many PTP uses and platforms
- Provide an extensible application, which may hold extensions such as white rabbit
The IEEE 1588 standard defines the data structures to be implemented by any software implementation

A PTP implementation is PTPd, which is practically the reference implementation

The white rabbit software implementation (ptp-noposix) is a fork of PTPd and made it runnable on WR switch and WR node
PTPd vs PPTP

- PTPd code layout is “event oriented”
- Source code uses CamelCase notation
- Platform specific implementation is mixed to algorithm code
- Transport on UDP is the only available transport mechanism (and it's not trivial to add ethernet transport)

- PPTP code layout is “state machine oriented”
- Source code uses CamelCase only where IEEE standard defines struct names
- Platform implementation is well separated and permits new architectures to be easily defined
- Many transport layers can be developed (e.g. raw ethernet)
$ ls -l
total 116
drwxr-xr-x 3 colosimo colosimo 4096 Mar  3 12:55 arch-bare-linux
drwxr-xr-x 3 colosimo colosimo 4096 Mar 23 07:08 arch-gnu-linux
drwxr-xr-x 4 colosimo colosimo 4096 Mar  3 12:55 arch-spec
drwxr-xr-x 2 colosimo colosimo 4096 Mar 23 07:08 diag
-rw-r--r-- 1 colosimo colosimo 1557 Mar  9 08:30 fsm.c
drwxr-xr-x 3 colosimo colosimo 4096 Jan  9 08:44 include
drwxr-xr-x 2 colosimo colosimo 4096 Mar 23 07:08 lib
drwxr-xr-x 2 colosimo colosimo 4096 Mar  9 09:00 proto-ext-whiterabbit
drwxr-xr-x 2 colosimo colosimo 4096 Mar 23 07:08 proto-standard
drwxr-xr-x 2 colosimo colosimo 4096 Mar 23 07:10 tools

$ ls -l diag
total 36
-rw-r--r-- 1 colosimo colosimo 1113 Jan  9 09:10 diag-no.c
-rw-r--r-- 1 colosimo colosimo  607 Jan  9 08:44 diag-printf.c
-rw-r--r-- 1 colosimo colosimo 1349 Mar  9 08:30 diag-yes.c
-rw-r--r-- 1 colosimo colosimo 13871 Jan  9 08:44 printf-full.c
-rw-r--r-- 1 colosimo colosimo  1360 Jan  9 08:37 printf-mini.c
int pp_state_machine(struct pp_instance *ppi, uint8_t *packet, int plen)
{
    struct pp_state_table_item *ip;
    int state, err;

    if (packet)
        msg_unpack_header(ppi, packet);

    state = ppi->state;

    /* a linear search is affordable up to a few dozen items */
    for (ip = pp_state_table; ip->state != PPS_END_OF_TABLE; ip++) {
        if (ip->state != state)
            continue;
        /* found: handle this state */
        ppi->next_state = state;
        ppi->next_delay = 0;
        pp_diag_fsm(ppi, ip->name, STATE_ENTER, plen);
        err = ip->f1(ppi, packet, plen);
        if (!err && ip->f2)
            err = ip->f2(ppi, packet, plen);
        if (err)
            pp_diag_error(ppi, err);
        pp_diag_fsm(ppi, ip->name, STATE_LEAVE, 0 /* unused */);

        ppi->is_new_state = 0;

        /* done: accept next state and delay */
        if (ppi->state != ppi->next_state) {
            ppi->state = ppi->next_state;
            ppi->is_new_state = 1;
        }
        return ppi->next_delay;
    }

    /* Unkown state, can't happen */
    pp_diag_error_str2(ppi, "Unknown state in FSM", "]
return 10000; /* No way out. Repeat message every 10s */
ptpdump: a tool for ptp message logging

TIME: (1331283886 - 0x4f59c7ae) 10:04:46.141962
ETH: 0800 (1c:6f:65:30:dd:61 -> 01:00:5e:00:01:81)
IP: 17 (192.168.0.253 -> 224.0.1.129) len 72
UDP: (319 -> 319) len 52
VERSION: 2 (type 0, len 44, domain 0)
FLAGS: 0x0002 (correction 0x00000000)
PORT: 1c-6f-65-ff-fe-30-dd-61-00-01
REST: seq 0, ctrl 0, log-interval 0
MESSAGE: (E) SYNC
MSG-SYNC: 1331283886.141867485
DUMP: payload (size 44)
DUMP: 80 02 00 2c 00 00 02 00 00 00 00 00 00 00 00
DUMP: 00 00 00 00 1c 6f 65 ff fe 30 dd 61 00 01 00 00
DUMP: 00 00 00 00 4f 59 c7 ae 08 74 b9 dd

TIMEDELTA: 0 ms
TIME: (1331283886 - 0x4f59c7ae) 10:04:46.142242
ETH: 0800 (1c:6f:65:30:dd:61 -> 01:00:5e:00:01:81)
IP: 17 (192.168.0.253 -> 224.0.1.129) len 72
UDP: (319 -> 319) len 52
VERSION: 2 (type 0, len 44, domain 0)
FLAGS: 0x0002 (correction 0x00000000)
PORT: 1c-6f-65-ff-fe-30-dd-61-00-01
REST: seq 0, ctrl 0, log-interval 0
MESSAGE: (E) SYNC
MSG-SYNC: 1331283886.141867485
DUMP: payload (size 44)
DUMP: 80 02 00 2c 00 00 02 00 00 00 00 00 00 00 00
DUMP: 00 00 00 00 1c 6f 65 ff fe 30 dd 61 00 01 00 00
DUMP: 00 00 00 00 4f 59 c7 ae 08 74 b9 dd
Current status

- The code infrastructure which permits multiplatform compilation and extensions is available on ohwr.org
- The feature aset of PTPd is fully implemented and tested on GNU Linux architecture
- In addition to PTPd, PPTP can now handle transport over raw ethernet sockets
- The implementation on other platforms (e.g. SPEC) is being worked on
- The implementation of white rabbit extension is still missing
Open issues

- The current WR PTP is a WR specific code base; how difficult is it to make it an optional extension?

- Is our current source code really open to any extensions? Will other extension have different requirements?
The first step of the project (replicate PTPd in our developing environment) was achieved.

Next two steps are:

- Make pptp available on bare linux and SPEC platform (an expert developer already demonstrated that this can be achieved in a few hours).
- Implement white rabbit extension, as an optional feature to be selected at compile and/or runtime.
- Source code is available at:
  git@ohwr.org:white-rabbit/pptp.git